Summary

This report presents the findings from a penetration testing exercise conducted on the basic web challenge available at HackThisSite.org (https://www.hackthissite.org/missions/basic/). It explores the ways in which web application vulnerabilities can be exploited and demonstrates how such exploits could potentially compromise a web application. This could lead to breaches in the CIA triad which might result in significant business, financial, or reputational damage to an organization. The aim of this report is to uncover vulnerabilities in the web application, attempt to exploit them, and offer recommendations for enhancing its security.

Mission 1: Hidden Password in HTML Comment

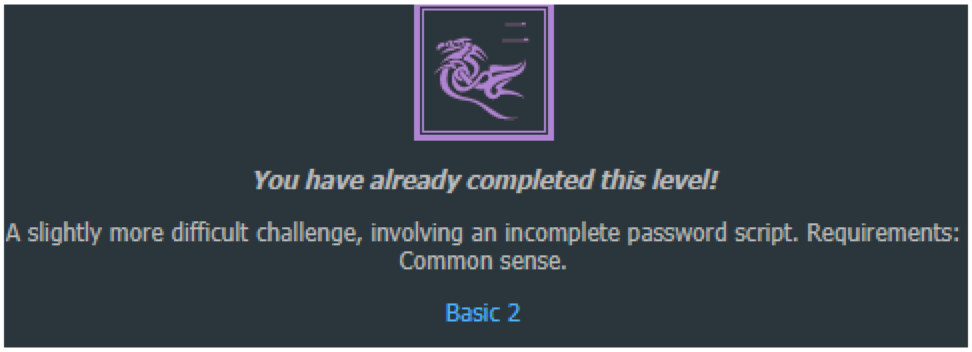
You have already completed this level!

Basic test of your skills to see if you can do any of these missions. Requirements: HTML

Basic 1

Overview: The challenge required finding a password hidden in an HTML comment.
Solution: The password was found in the HTML source code of the page.
Key Idea: Hidden information can often be found in the HTML comments or source code of a page.

Mission 2: Missing Password File



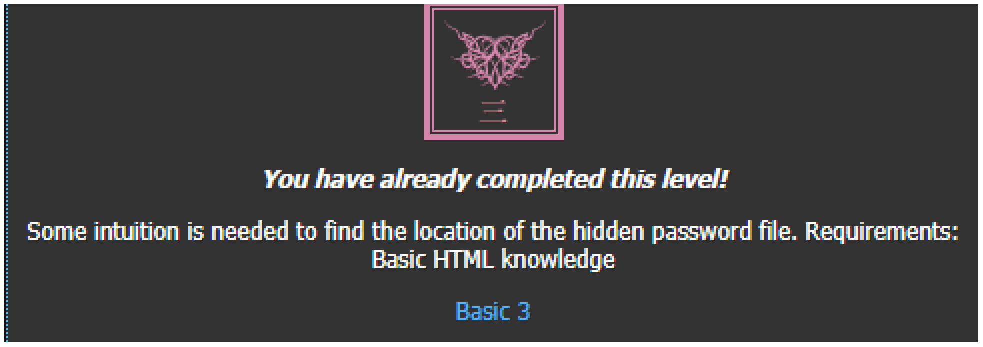
You have already completed this level!

A slightly more difficult challenge, involving an incomplete password script. Requirements: Common sense.

Basic 2

Overview: The password was expected to be read from a file that was not uploaded.
Solution: Submitting an empty input in the password form bypassed the authentication.
Key Idea: Test for missing files or improper handling of file dependencies.

Mission 3: Direct Access to Password File

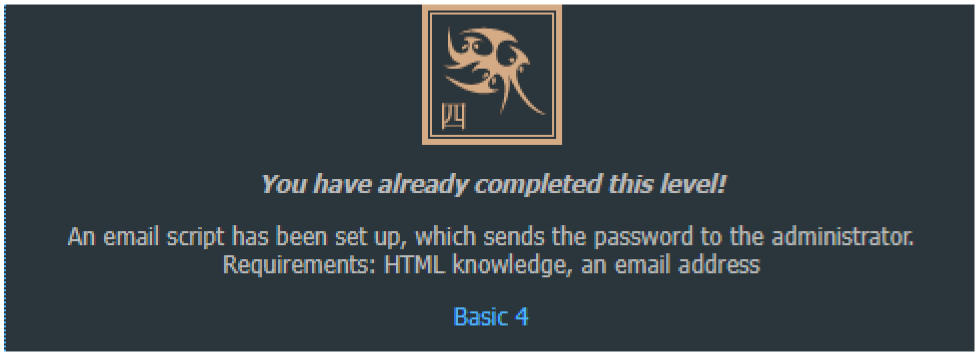

You have already completed this level!

Some intuition is needed to find the location of the hidden password file. Requirements: Basic HTML knowledge

Basic 3

Overview: The password file was accessible via a direct URL.
Solution: The password was obtained by navigating to the exposed file URL.
Key Idea: Always ensure sensitive files are not directly accessible through predictable URLs.

Mission 4: Hardcoded Password Sent via Email

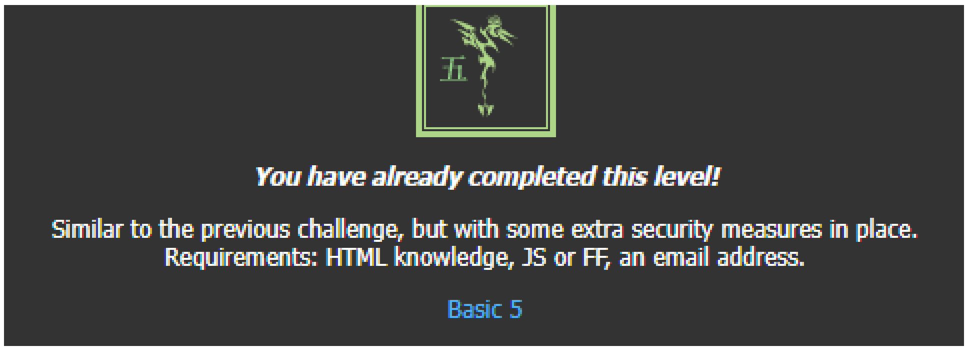

You have already completed this level!

An email script has been set up, which sends the password to the administrator. Requirements: HTML knowledge, an email address

Basic 4

Overview: The password was sent via email when a form was submitted.
Solution: Modified the email address in the HTML to receive the password.
Key Idea: Be cautious of exposing sensitive data through email and ensure input fields are properly sanitized.

Mission 5: Email-Based Password Retrieval

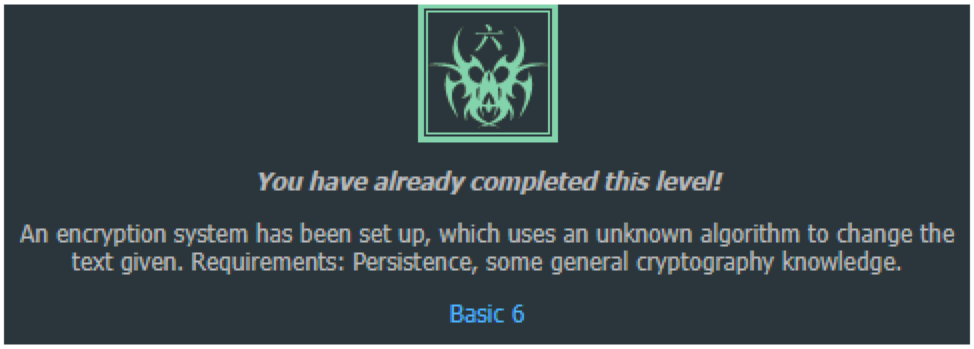

You have already completed this level!

Similar to the previous challenge, but with some extra security measures in place. Requirements: HTML knowledge, JS or FF, an email address.

Basic 5

Overview: The same vulnerability from Mission 4 was present.
Solution: Used the same method as Mission 4 to retrieve the password.
Key Idea: Persistent vulnerabilities should be addressed thoroughly to prevent reuse.

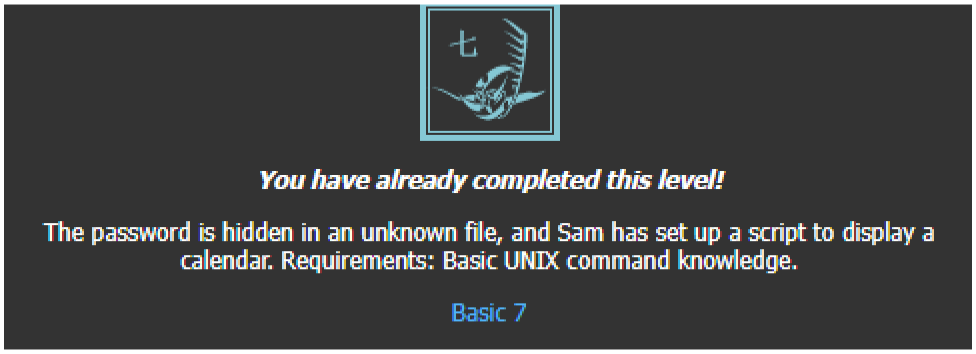
Mission 6: Encrypted Password


You have already completed this level!

An encryption system has been set up, which uses an unknown algorithm to change the text given. Requirements: Persistence, some general cryptography knowledge.

Overview: The password was encrypted using a simple shifting cipher.
Solution: Analyzed the encryption pattern and reversed it to retrieve the password.
Key Idea: Avoid using simplistic or easily reversible encryption methods.

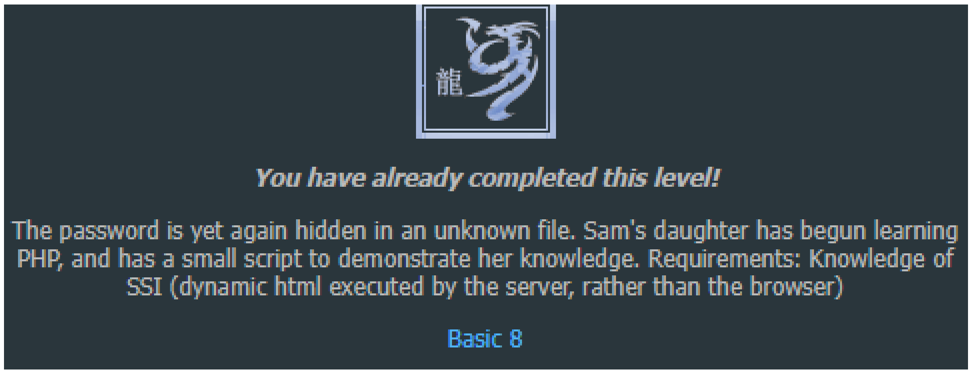
Mission 7: Directory Listing via Command Injection


You have already completed this level!

The password is hidden in an unknown file, and Sam has set up a script to display a calendar. Requirements: Basic UNIX command knowledge.

Overview: The challenge involved finding a file in a directory by injecting commands.
Solution: Used command chaining to list directory contents and locate the password file.
Key Idea: Protect against command injection by validating and sanitizing user inputs.

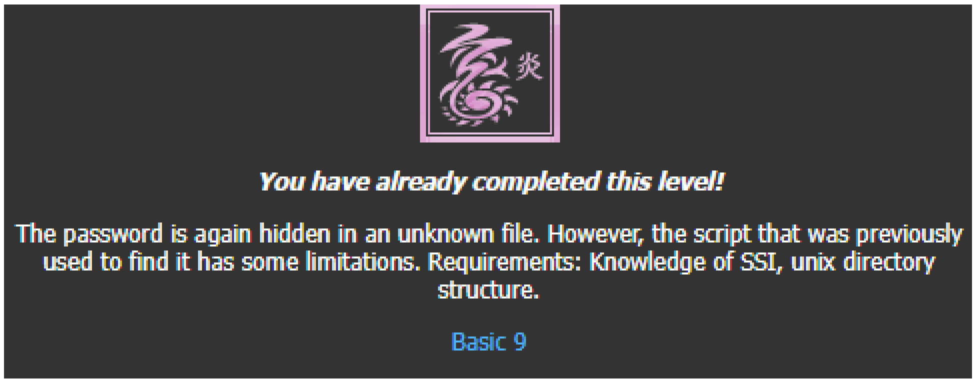
Mission 8: SSI Injection for Directory Listing


You have already completed this level!

The password is yet again hidden in an unknown file. Sam's daughter has begun learning PHP, and has a small script to demonstrate her knowledge. Requirements: Knowledge of SSI (dynamic html executed by the server, rather than the browser)

Overview: Used Server Side Includes (SSI) to list directory contents.
Solution: Injected SSI commands to reveal directory contents and locate the password file.
Key Idea: Server-side scripting can be abused if not properly configured or sanitized.

Mission 9: Directory Traversal and SSI Injection

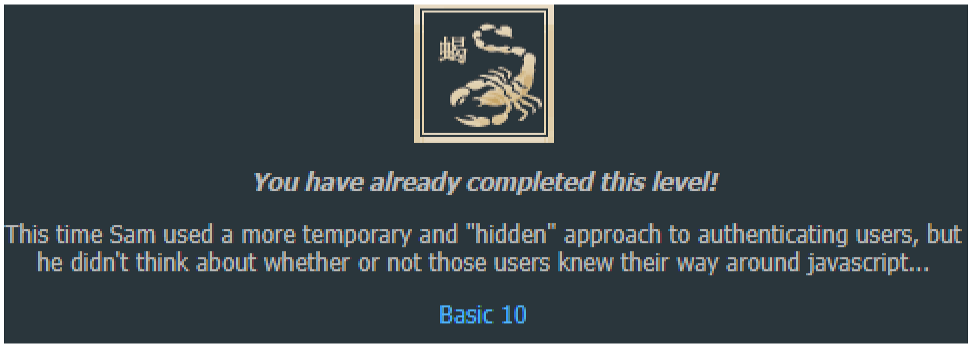

You have already completed this level!

The password is again hidden in an unknown file. However, the script that was previously used to find it has some limitations. Requirements: Knowledge of SSI, unix directory structure.

Basic 9

Overview: Similar to Mission 8 but required changing directories to find the password file.
Solution: Used SSI injection to traverse directories and locate the password file.
Key Idea: Directory traversal and SSI injection can expose sensitive data if not secured.

Mission 10: Cookie Manipulation

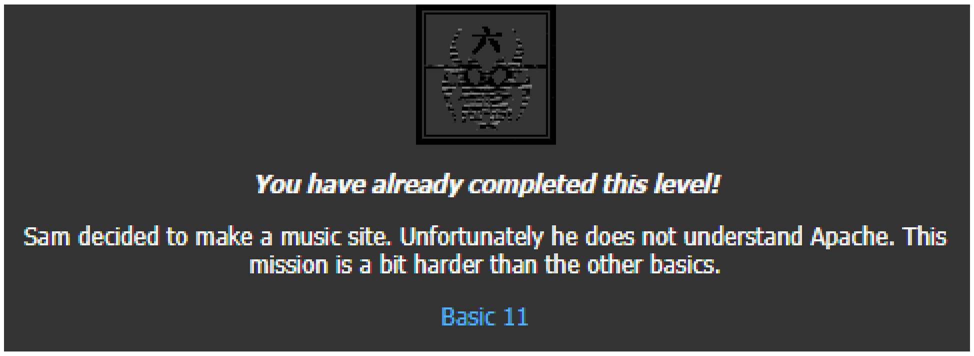

You have already completed this level!

This time Sam used a more temporary and "hidden" approach to authenticating users, but he didn't think about whether or not those users knew their way around javascript...

Basic 10

Overview: The challenge involved manipulating a cookie to bypass authentication.
Solution: Modified the document.cookie value to grant access.
Key Idea: Secure cookies and use proper validation to prevent unauthorized access.

Mission 11: Directory Listing and .htaccess Exposure


You have already completed this level!

Sam decided to make a music site. Unfortunately he does not understand Apache. This mission is a bit harder than the other basics.

Basic 11

Overview: Exploited directory listing and visible .htaccess file to find a hidden password.
Solution: Navigated through directories and accessed the .htaccess file to find clues about the password.
Key Idea: Disable directory listing and protect configuration files from public access.

Recommendations

- Improve Password Management: Avoid hardcoding passwords and ensure that sensitive information is not exposed through HTML comments or unprotected files.
- Sanitize Inputs: Implement strict input validation to prevent injection attacks and unauthorized file access.
- Secure Encryption: Use strong, modern encryption methods instead of simple ciphers and ensure encryption keys are securely managed.
- Proper File Handling: Ensure that files containing sensitive information are not accessible via predictable URLs and are properly protected.
- Secure Cookies: Use secure attributes for cookies (e.g., HttpOnly, Secure) and validate cookie values to prevent tampering.
- Server Configuration: Disable directory listing and restrict access to configuration files. Also ensure server-side scripts are securely configured.